



LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

Pythia Jet Finding Study with Trento Backgrounds

J. Simpson, R. Soltz

June 22, 2016

Disclaimer

This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Pythia Jet Finding Study with Trento Backgrounds

Joseph Simpson^{a,b} and Ron Soltz^b

^a *United States Naval Academy, Annapolis, Maryland 21402, USA*

^b *Lawrence Livermore National Laboratory, Livermore California 94550, USA*

Abstract

We present results applying the Pythia SlowJet Finder to Pythia generated QCD and QED hard processes in the presence of simulated heavy ion backgrounds. The hard process events are generated with Pythia version 8.219 for $\sqrt{s_{NN}}=200$ GeV proton-proton collisions and the backgrounds are generated by the Reduced Thickness Event-by-event Nuclear Topology model T_RENTo for Au-Au collisions with a nucleon-nucleon cross-section of 4.23 fm². The T_RENTo model is used to calculate the initial entropy and ellipticity from which the total charged particle multiplicity and elliptic flow are determined. We report results in the form of event displays, total p_T distributions, and fragmentation distributions for SlowJet applied to Pythia events with and without the simulated heavy ion backgrounds.

Contents

1	Introduction	2
2	Running Pythia	2
3	Adding Trento Backgrounds	3
4	Working with SlowJet Finder	5
5	Studying Jets	7
A	Listing of Python Scripts	11

1 Introduction

Motivation

The study of jet-quenching in heavy ion collisions, how jets lose energy as they evolve within the quark-gluon plasma (QGP) is one of the most important topics remaining in the quest to understand the detailed properties of the QGP. However, the task of reconstructing jets within a high multiplicity heavy ion background is a challenge for RHIC energies. Fluctuations in the background can be falsely reconstructed as jets, and true jets that lose energy in the plasma can easily be buried within the backgrounds. The purpose of this study is to develop a set of simple tools to study jets within an environment that simulates the basic elements of a heavy ion background with thermal particle production and radial and elliptic flow, and then to use these tools to create visual images of the jets and quantitative distributions of jet properties in a fluctuating heavy ion background. For this study, it is assumed that the identity, mass, and momenta of all produced particles are precisely known. We do not attempt to include any detector effects in our simulation.

Software Framework

For this study we work with the Pythia 8.219 Python package, which can be called from Python (using v2.7), and we use the SlowJet routines that are included in the Pythia installation. SlowJet provides an option to run an anti- k_T algorithm that produces identical results to FastJet, but does not run as quickly. Most of the analysis is done using numpy and matplotlib.pyplot packages. We use T_RENTo version 1.3 code to create heavy ion backgrounds. The T_RENTo code is run as a stand-alone program. T_RENTo outputs are saved to text files and read from Python scripts, although it is also possible for it to be called directly from within Python. Unless stated otherwise, all T_RENTo backgrounds are for minimum bias distributions. Additional details on using these packages are given in the descriptions below.

2 Running Pythia

Pythia allows users to set the beam center-of-mass energy (E_{CM}). By default Pythia sets E_{CM} to 14 TeV, the default for all our programs is 200 GeV. Initial QCD or QED hard scattering processes may be turned on or off. Pythia also allows restrictions to be set on the transverse momentum produced by hard scattering ($p_{T_{Hat}}$ or jet p_T). For our programs the jet p_T range is typically 20-25 GeV. Figure 1 shows the distribution of jet p_T for various QCD and QED processes for jets above 20 GeV. The upper histogram of Figure 1 shows the four most common QCD hard processes produced by Pythia. The different processes are shown in the legend. Gluons are represented by "g", quarks (uds) are represented by "q", and anti-quarks (\bar{uds}) are represented by "q(bar)". There are also other hard processes, such as those that involve heavy quarks, but they occur so infrequently that they are not included. The lower histogram shows all five of the QED hard processes, however only two of the processes occur frequently. For these processes photons are represented by "gamma," and fermions participating in electroweak interactions are represented by "f" and "f(bar)".

Figure 2 shows two lego plots, one for a QCD event and one for a QED event. These plots

demonstrate how our programs are able to display each Pythia event. Each particle that is produced from a Pythia event is binned in a 2 dimensional histogram depending on its pseudo-rapidity (η) and azimuthal angle (ϕ). The histogram bins are weighted by transverse energy (E_T). In both plots it is easy to identify the jets produced by the Pythia events. For the QED plot it is important to note that one of the jets is produced by only a single photon.

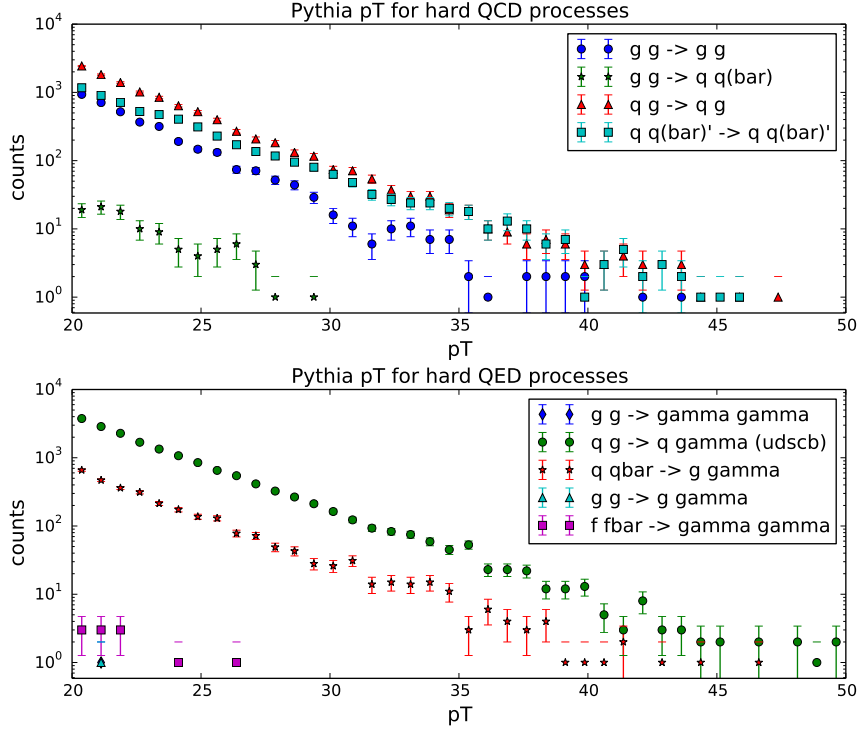


Figure 1: Distribution of jet pT for QCD and QED processes for 20,000 events. Figure created with [python pythia_pT_process.py -m 20000]

3 Adding Trento Backgrounds

TRenTo allows users to input several physical options in order to generate initial states from which we are able to derive realistic heavy ion collision backgrounds. These options include two projectiles, number of events, reduced-thickness, fluctuation, nucleon-width, cross-section, and a normalization factor. For each data set, we used non-deformed gold (Au) nuclei with 197 nucleons each. We produced 100,000 events for several different cross-sections. All other physical options were left to their default settings: reduced-thickness = 0, fluctuation = 1, nucleon-width = 0.5 fm, and normalization = 1. Different cross-sections correspond to different beam E_{CM} per nucleon pair. The cross-sections (in units of fm²) that we used to simulate the different beam energies

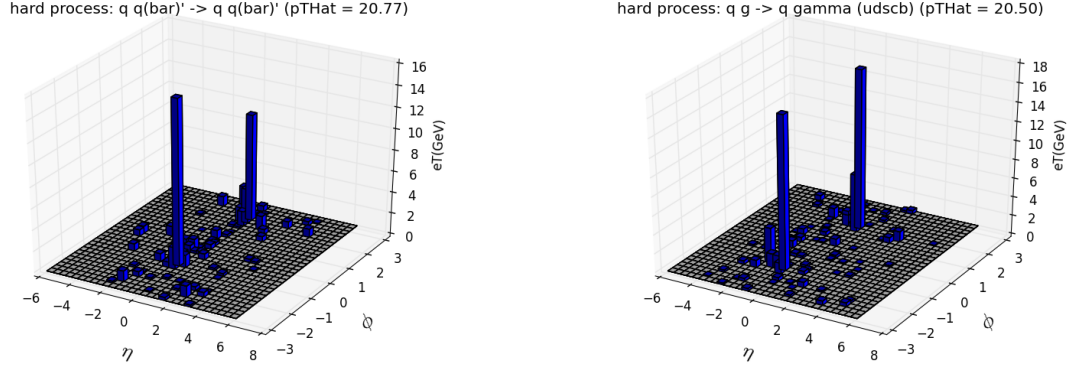


Figure 2: Pythia event display for a QCD event, shown to the left, and a QED event shown to the right. Left figure created with `[python 2d_pythia_slowjet.py -o -b 30]`. Right figure created with `[python 2d_pythia_slowjet.py -o -c -q -b 30]`

are shown in Table 1. An important note is that the corresponding cross-section for 14.5 GeV was shown corresponding to 15.0 GeV for the PHENIX data, however we assumed this be a typo. T_RENTo reads in the options and produces a text file listing several event properties for each event: event number, impact parameter, number of participants, initial entropy, and properties of flow. The produced T_RENTo text files were then imported into our programs as an array of event data. To use the T_RENTo data as we intended we had to produce accurate Au-Au multiplicities for the

Table 1: Cross Sections and Scaling Constants

$E_{CM}(\text{GeV})$	$\sigma_{nn}(\text{fm}^2)$	Scaling Constant
200	4.23	4.66
130	3.96	3.97
62.4	3.60	3.13
39	3.43	2.55
27	3.32	2.16
19.6	3.25	1.95
14.5	3.20	1.65
7.7	3.12	1.27

various beam energies. To do this we fit the T_RENTo data to PHENIX data as shown in Figure 3. We created a histogram for the number of participants in each T_RENTo event and weighted the bins by the initial entropy. We then scaled the T_RENTo data at each energy by a constant to fit it to the PHENIX data of charged multiplicity vs. number of participants. The T_RENTo data points are the open shapes that and are connected by lines. Using the scaling constants from the fit at each energy allowed us to create multiplicity values from the initial entropy values. The scaling constants that we used are shown in Table 1.

Using the T_RENTo multiplicity values we generated background particles for each T_RENTo event. To generate values of transverse momentum (p_T), pseudo-rapidity (η), and azimuthal angle (ϕ) for each particle we used the numpy random number generator. For ϕ and η we created flat distributions between $-\pi$ to π and -2 to 2 , and for p_T we sampled from a Gaussian deviate, which is equivalent to a thermal Boltzmann distribution. We also included radial and elliptic flow into our background distributions by using the ϵ_2 variable from the T_RENTo data. First we calculated the initial transverse rapidity (y_T) and then applied flow as an additive boost.

$$y_{Boost} = \rho_0 r + \rho_2 \epsilon_2 \cos(2\phi)$$

The scaling parameter for radial flow (ρ_0) that we used was 0.85, taken from Table 2 of [1]. The scaling parameter for elliptic flow (ρ_2) was 0.15 from Figure 4 of [2]. Using the numpy random number generator we created a normalized radial distance (r). After applying flow, y_T was converted back to p_T . The mass used for these calculations was chosen randomly from one of nine different particles that were included in the produced background. These particles consisted of pions, kaons, protons, and neutrons, as well as their anti particle counterparts. Figure 4 shows the background display produced from T_RENTo events. Again each particle that is produced is binned in a 2 dimensional histogram in (η, ϕ) space. For this plot the histogram bins are weighted by p_T . There are no jets produced within the T_RENTo background.

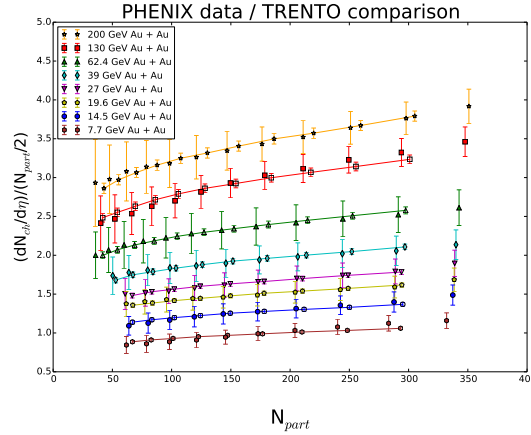


Figure 3: Fit of T_RENTo data to PHENIX data. Figure created with [python phenix_trento.comp.py]

4 Working with SlowJet Finder

The SlowJet jet finder takes several input parameters that affect how it searches for jets. These parameters are *power*, *radius*, *p_TjetMin*, *etaMax*, *select*, and *massSet*. The *power* parameter determines which jet finding algorithm to use. For our programs we set *power* to -1, this corresponds to the *anti-k_T* algorithm. Other possible algorithms for SlowJet are the Cambridge/Aachen

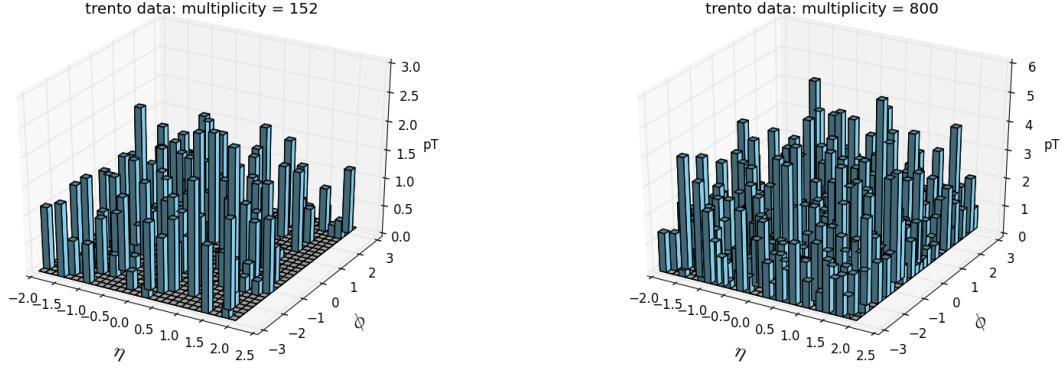


Figure 4: T_RENTo event display (no jets). High multiplicity is shown on the right and lower multiplicity on the left. Both plots created with `[python 2d_trento.py -b 30]`

algorithm and the k_T algorithm. The *radius* parameter is crudely related to the radius of the jet cone in (y, ϕ) space. In most of our programs *radius* is set to 0.7 by default. The $p_{T,jetMin}$ parameter is the minimum p_T that a cluster must have to become a jet. Our values for $p_{T,jetMin}$ are normally set to 10 or 15 GeV. The *etaMax* parameter is the maximum \pm pseudo-rapidity that the detector is assumed to cover. For our programs *etaMax* is always set to 2. The *select* parameter determines which particles are analyzed by SlowJet. For our programs *select* is always set to 2 which corresponds to all observable final-state particles. The *massSet* parameter determines the masses assumed for the particles analyzed by SlowJet. There are three *massSet* options: 0 for all massless, 1 for photons are massless while all others are assigned the mass of charged pions, and 2 for all given their correct masses. For our programs *massSet* is set to 2, with an option for it to be set to 1. The *radius* and $p_{T,jetMin}$ parameters are most often changed while we examine jets.

The *anti- k_T* algorithm functions by finding the hardest (highest p_T) particle in an event and grouping all soft particles within a certain radius into a cluster. This is then repeated for the next hard particle and so on. If a hard particle has no hard neighbors within a distance $2R$ then it will group all soft particles within a distance R into its jet cone. If two hard particles are within a distance R they will be grouped into the same cluster. If the two hard particles are further apart than R but within a distance $2R$, the hard particle with the higher p_T will have a complete cone that cuts into the cone of the other. If the p_T of the two hard particles is very close then the boundary between the cones will be a straight line. In this way the *anti- k_T* algorithm is resilient with respect to soft radiation but flexible with respect to hard radiation. In the language of jet physics, this is referred to as being infrared and collinear safe. See [3] for more on jet finding algorithms including *anti- k_T* .

Figure 5 and Figure 6 demonstrate how well SlowJet is able to function in Pythia events with varying amounts of heavy ion background. Both figures use the same familiar event display that was shown in Figure 2 and Figure 4 and use the standard SlowJet settings discussed above. In these figures the color scheme is that Pythia jets that are identified correctly by SlowJet are colored

green, particles mistakenly identified as SlowJet jets are colored yellow, Pythia jets that are missed by SlowJet are colored red, and all other background particles are colored blue. Figure 5 shows two Pythia events with no heavy ion background. In the event shown on the left SlowJet correctly identifies both jets. However, in the event on the right, the second jet is too diffuse and SlowJet does not identify it as a jet. Figure 6 is a greater test for SlowJet because now a T_RENTo background is introduced. In the low T_RENTo multiplicity event, SlowJet is still able to correctly identify the two jets, however a few background particles, shown in yellow, are also lumped in with the jets. In the high T_RENTo multiplicity event SlowJet becomes confused. In this case there are many T_RENTo particles identified as jets, those shown in yellow. The true jets become lost in the background and SlowJet fails.

All the bar plots in Figure 5 and Figure 6 are created in a way so that bars are stacked on top of each other if two different color bars share the same bin. The order in which the bars are stacked is that the blue bars corresponding to background not mistaken as jets goes on the bottom. The red bars associated with true jet particles not identified as jets are stacked next. Then the yellow bars associated with falsely identified jets are stacked. Correctly identified jets, shown in green, are then stacked on the very top. This order of stacking the bars is the most useful when displaying very congested events because it is easy to see how SlowJet performed.

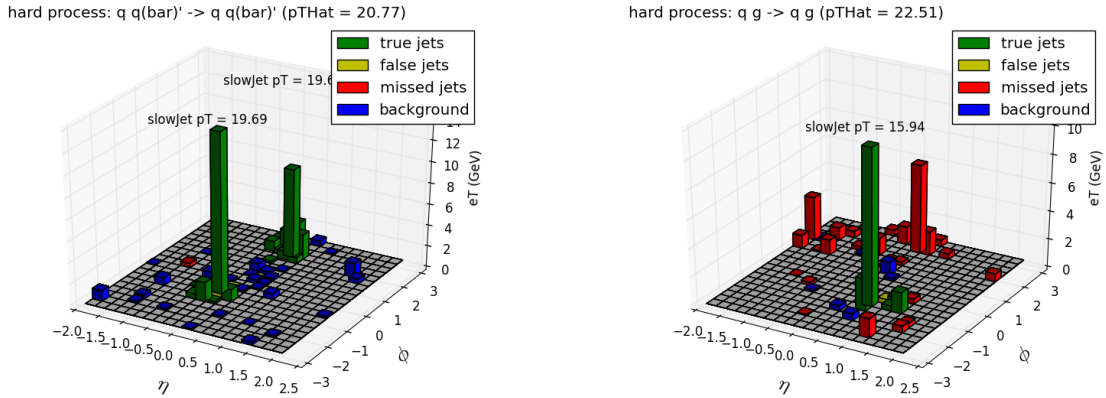


Figure 5: Event display for Pythia with SlowJet Finder, no heavy ion background. SlowJet succeeds on the left but struggles on the right. Both plots created with [python 2d_pythia_slowjet_truejet.py]

5 Studying Jets

To study jets and explore the different parameters of SlowJet we first focused on the SlowJet parameters of *radius* and *p_{TjetMin}*. One of our programs examined how many jets were identified in Pythia events with T_RENTo backgrounds. Ideally SlowJet should identify 2 jets for each Pythia event. We found that when *p_{TjetMin}* was set low and *radius* was set high, far too many jets would be identified on average. However, if *p_{TjetMin}* was set high and *radius* set low, very few

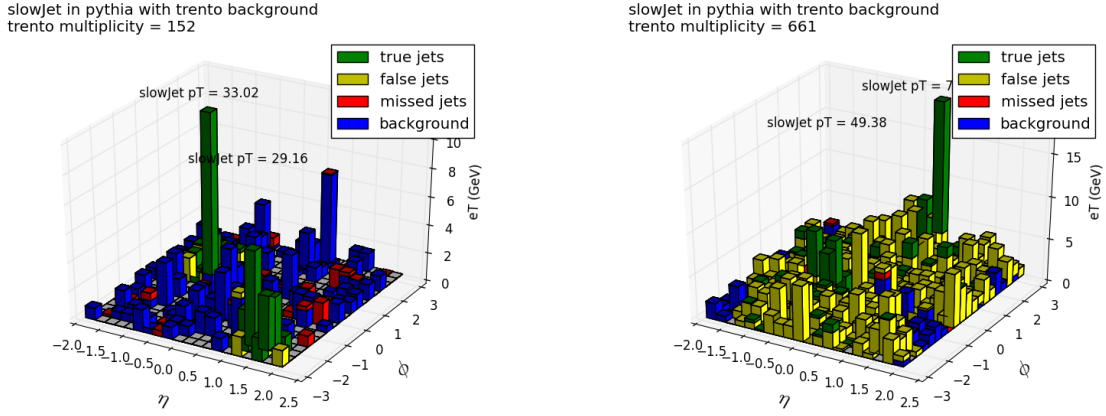


Figure 6: Event display for Pythia+Trento with SlowJet Finder. Low multiplicity $T_{\text{RENT}}\text{O}$ background shown to the left and high multiplicity $T_{\text{RENT}}\text{O}$ background shown to the right. Correctly identified jets are colored green, mistakenly identified jets are colored yellow, jets that SlowJet missed are colored red, and all background particles are colored blue. Both plots created with [python 2d_pythia_trento_slowjet_truejet.py]

jets would be identified. In the $T_{\text{RENT}}\text{O}$ backgrounds it is difficult to raise the requirements for clusters to be identified as jets while still being able to identify all Pythia jets. The plots produced by this program are shown in Figure 7. Another program restricted the p_T of Pythia jets to 20-25 GeV and set $p_{T\text{jetMin}}$ to 15 GeV. The $radius$ parameter was then varied to see if the identified jets would be in the correct p_T range. We found similar results. Increasing $radius$ to values as high as 0.8 caused the peak of identified jets to be within the proper p_T range but also allowed many jets above the range to be identified as well. Decreasing $radius$ to values as low as 0.3 caused the peak of identified jets to fall below the proper p_T range but reduced the number of jets that were identified above the range.

Figure 8 shows the effect on the SlowJet finder of adding $T_{\text{RENT}}\text{O}$ background to Pythia events. Both plots in Figure 8 use SlowJet with our default settings. In the plot on the left SlowJet is able to calculate the correct jet p_T on average. However, in the plot on the right with $T_{\text{RENT}}\text{O}$ background included, the SlowJet p_T values are higher than they should be. This occurs because SlowJet is incorrectly identifying $T_{\text{RENT}}\text{O}$ background particles as parts of jets.

Our next focus was on the fragmentation function value (ξ) for particles within both SlowJet jets and true jets. ξ provides information on how the energy is distributed along the jet. To calculate ξ we used the following formula:

$$\xi = \ln((p_{\text{jet}}^{\rightarrow} \cdot p_{\text{jet}}^{\rightarrow}) / (p_{\text{jet}}^{\rightarrow} \cdot p_{\text{prt}}^{\rightarrow}))$$

In Figure 9, ξ is calculated for both QCD and QED events with no $T_{\text{RENT}}\text{O}$ data on the left and with $T_{\text{RENT}}\text{O}$ data on the right. ξ is calculated for SlowJet jets with $radius$ set at 0.3, 0.5, and 0.7 in all four plots. In the plots it is noticeable that as $radius$ is decreased the peak of ξ values for reconstructed jets is reduced. This occurs because particles with higher ξ values are

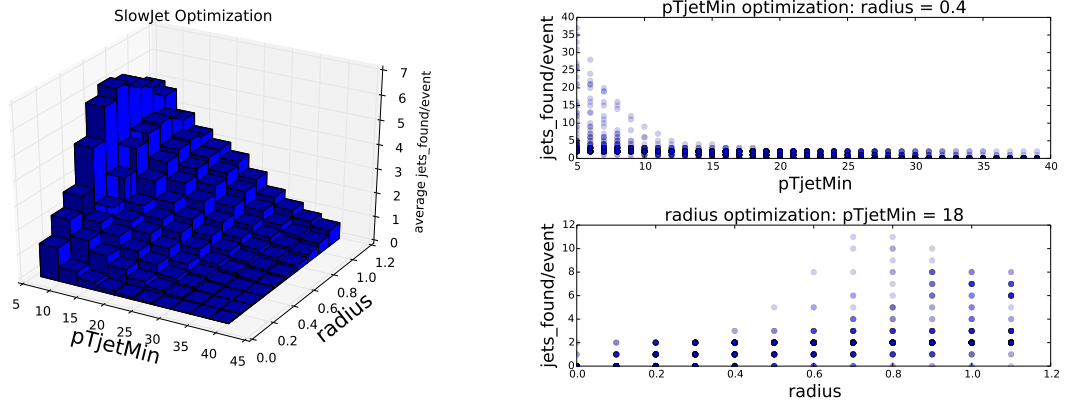


Figure 7: The 2 dimensional histogram on the left shows the average number of jets found by SlowJet in Pythia events with T_RENTO background. The plots on the right are 1 dimensional scatter plots that explore the optimization of each parameter individually. All plots created with [python slowjet_pTmin_radius.py]

farther away from the center of the jet cone. As *radius* is decreased particles on the edge of the jet cone are no longer included in the reconstructed jets. True jets consist of all particles that are ancestors of the initial hard scattering in each Pythia event. These particles were identified by tracing the daughters of each particle starting with the two initial particles that participated in the hard process and ending when only final state particles remain.

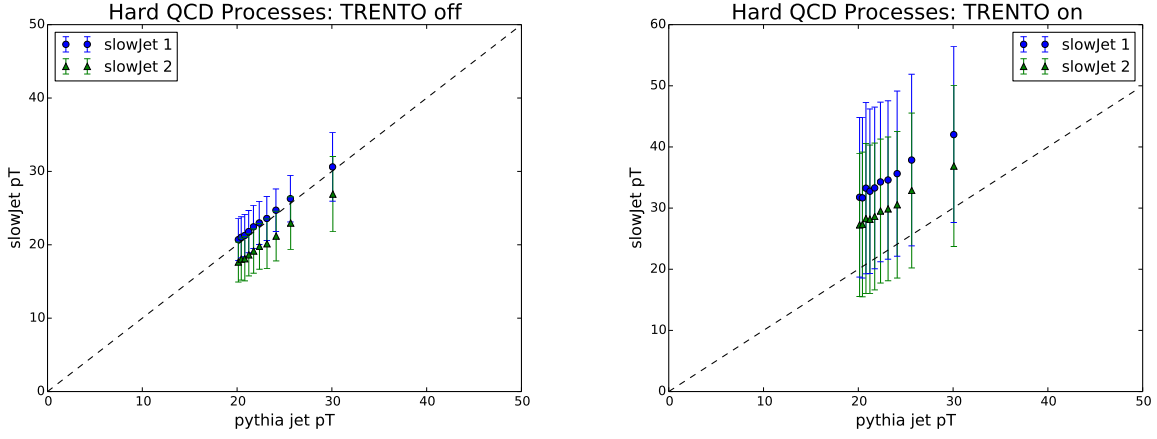


Figure 8: The left plot shows SlowJet p_T vs. the true p_T of Pythia jets without T_RENTo background. The right plot has T_RENTo background included. The left plot was created with `[python slowJetpT_vs_pTHat.py -i -u 10000]`. The right plot was created with `[python slow-JetpT_vs_pTHat.py -t -i -u 10000]`.

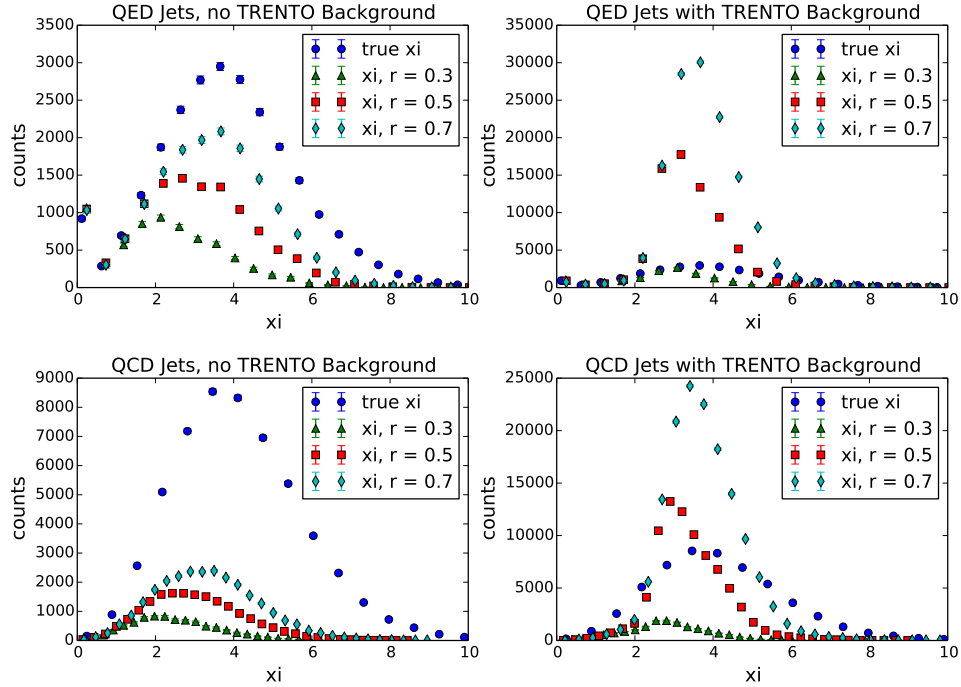


Figure 9: Fragmentation function value (ξ) distributions for true and reconstructed QCD and QED jets, with and without trento backgrounds. Figure created with `[python xi_slowjet_truejet.py]`

A Listing of Python Scripts

The following scripts were used to create the figures shown in this document.

2d_pythia_slowjet.py Displays Pythia events in (η, ϕ) space, SlowJet jets are colored red

2d_trento.py Displays T_RENTo backgrounds in (η, ϕ) space

2d_pythia_slowjet_truejet.py Displays Pythia events in (η, ϕ) space, SlowJet jets are compared to true jets

2d_trento_slowjet.py Displays T_RENTo background in (η, ϕ) space, SlowJet identified jets are colored red

2d_pythia_trento_slowjet.py Displays Pythia events with T_RENTo backgrounds in (η, ϕ) space, bars are colored to show SlowJet performance

2d_pythia_trento_slowjet_truejet.py Displays Pythia events with T_RENTo backgrounds in (η, ϕ) space, SlowJet jets are compared to true jets

3d_pythia_slowjet.py Displays Pythia events in (η, ϕ) space, particles shown individually, SlowJet jets are colored red

phenix_data.py PHENIX data plotted with error bars for charged multiplicity vs. number of participants, Au-Au at various energies

phenix_trento_comp.py Fits T_RENTo data to PHENIX data for various Au-Au energies, charged multiplicity vs. number of participants

pythia_mult_process.py Plots frequency of charged multiplicities for various QED/QCD Pythia hard processes

pythia_pT_process.py Plots frequency of p_T values for various QED/QCD Pythia hard processes

scanPythia.py Prints SlowJet information about Pythia events such as jet constituents and jet p_T

slowJetpT-vs-pTHat.py Compares the true jet p_T produced by Pythia QCD/QED events to the SlowJet jet p_T

slowjet_pTmin_radius.py Creates a 2d histogram for SlowJet parameters $radius$ and $p_{TjetMin}$ and shows average jets found for various values

slowjet_pTrange_radius.py Tests various values of SlowJet $radius$ while the Pythia jets are in a restricted p_T range

trento_mult_npart.py Plots initial entropy (mult) vs. number of participants (Npart) for T_RENTo data

trento_plot_values.py Creates plots for T_RENTo data, such as initial entropy vs. impact parameter and number of participants vs. impact parameter, as well as others

trento_phi_pT_eta.py Plots the values of ϕ , p_T , and η that are created for the T_RENTo backgrounds

xi_slowjet.py Creates a histogram of ξ values for SlowJet jets

xi_truejet.py Creates a histogram of ξ values for true jets

xi_slowjet_truejet.py Compares ξ for true and reconstructed jets

References

- [1] F. Retiere and M. Lisa, *Phys. Rev. C* 70, 044907 (2004).
- [2] B. Alver and G. Roland, *Phys. Rev. C* 81, (2010).
- [3] M. Cacciari, G. P. Salam, and G. Soyez, *Eur. Phys. J. C* 72, 1 (2012).